YIOOP! INTRODUCING AUTOSUGGEST AND SPELL CHECK

A Project Report

Presented to

The faculty of Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Sandhya Vissapragada

December 2012

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

YIOOP! INTRODUCING AUTOSUGGEST AND SPELL CHECK

by

Sandhya Vissapragada

APPROVED FOR THE DEPARTMENT OF

COMPUTER SCIENCE

Dr. Chris Pollett, Department of Computer Science Date

Dr. Sami Khuri, Department of Computer Science Date

Dr. Robert Chun, Department of Computer Science Date

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research Date

ABSTRACT

YIOOP! INTRODUCING AUTOSUGGEST AND SPELL CHECK

This project adds autosuggest and spell-check for queries in Yioop [1], a PHP-based search engine. These features help a user by reducing typing, by catching any spelling errors, and by making it easier to repeat searches. Commercial search engines like Google, run on machine clusters and use lists of popular queries from their logs to provide relevant suggestions to users. Efficient storage of data on multiple servers is responsible for minimizing response times. Yioop typically runs on a smaller number of machines compared to commercial search engines. This project aims to implement these computationally intensive functionalities in this constrained environment. This is achieved by performing any needed processing on the client-side without sending queries to the Yioop server.

ACKNOWLEDGEMENTS

This project is the result of the efforts of many individuals. I am very thankful to my advisor, Dr. Chris Pollett, for his guidance and constant encouragement throughout the project. I would also like to thank my committee members, Dr. Sami Khuri and Dr. Robert Chun, for their time and effort. Finally, I would like to thank my family and friends, for their help, encouragement and constant support.

TABLE OF CONTENTS

1. Introduction 8
2. Yioop Search Engine 10
3. Preliminary Work 11
    3.1 Dictionary Words 12
    3.2 Storing Dictionary Words 12
    3.3 Timing Tests 15
    3.4 Google Autosuggest 17
    3.5 Performance Of Loading The Dictionaries 19
4. Autosuggest In Yioop 20
5. Multi-Word Suggest 22
6. Foreign Language Support 23
7. Using Local Storage To Suggestions 25
    7.1 An Algorithm For Local Storage Suggestions 25
8. Spell Correction 27
    8.1 Edit Distance Algorithm For Spell Correction 27
    8.2 Evaluation 29
9. Suggestions For Transliterated Queries 30
10. Performance 32
11. Conclusion & Future Work 35
12. References 36

LIST OF FIGURES

Figure 3.1 Example of a trie 12

Figure 3.2 Structure of a generated trie 13

Figure 3.3: Load time of a compressed trie sent in a text file 15

Figure 3.4: Load time of a zipped trie with a browser enabled to accept gzip files 17

Figure 3.5: A Google search page being inspected in a Safari activity window 18

Figure 3.6: A Google search page being inspected in a Safari activity window for the query “Soc”18

Figure 3.7: A downloaded file with data on auto-suggest results for the query “Soc” 19

Figure 4.1: Yioop suggestion for the character ‘c’ 20

Figure 4.2: Yioop suggestion for the word ‘coll’ 21

Figure 4.3: Selecting a word from an auto-suggest list 21

Figure 5.1: Multi-word suggestion for the query “Social netw” 22

Figure 5.3: Using up/down arrow keys on a suggestion list 22

Figure 6.1 Suggestions for French words 23

Figure 6.2 Suggestions for French words 23

Figure 6.3 Suggestions for Russian words 24

Figure 7.1 Suggestions based on local storage 26

Figure 7.2 Suggestions based on local storage in the order of frequencies 26

Figure 8.1 Correction of the incorrect query ‘tha’ 28

Figure 9.1 Suggestions for Telugu term ‘కా’ transliterated as ‘kaa’ in English 31

Figure 9.2 Suggestions for Telugu term ‘లో’ transliterated as ‘loo’ in English 31

Figure 10.1 Comparison of multi-word suggest 33

Figure 10.1 Comparison of usage of previous queries 33

LIST OF TABLES

Table 3.1 A comparison of the load times of different formats of tries 16

Table 8.1 Experiment on spelling corrections using Edit distance 1 and 2 29

Table 10.1 Comparison of a user’s typing time with and without autosuggest 32

1. INTRODUCTION

Autosuggest, also called auto-complete, is a popular and useful feature provided these days in many web browsers, search engines and other web applications. It is usually provided as a dropdown menu of choices below the textbox in which the user is typing. The goal is to anticipate the text the user intends to type so that instead of typing in the rest of the query, one of the dropdown choices can be selected. Some advantages of autosuggest are queries need not be typed in their entirety, queries need not be remembered verbatim and incorrectly typed queries can be overridden without recomposing them.

This project aims to develop a similar autosuggest feature for Yioop, a PHP-based search engine. Commercial search engines like Google, run on machine clusters and use lists of popular queries from their logs to provide relevant suggestions to users. Efficient storage of data on multiple servers is responsible for minimizing response times. Yioop typically runs on a fewer machines compared to commercial search engines. This project aims to implement these computationally intensive functionalities in this constrained environment. This is achieved by performing any needed processing on the client-side without sending queries to the Yioop server.

The Yioop search engine is introduced in Section 2. Section 3 talks about the preliminary work that has been done to add autosuggest and spell-check features to Yioop. It describes the research conducted on various suggestion techniques, the data structures suitable for efficient storage of dictionary words and the network response times for sending dictionaries in different formats. Section 4 talks about how a basic autosuggest functionality was added to Yioop that handles just single word queries.

Multi-word suggestions are discussed in Section 5. Section 6 deals with supporting foreign language queries. We will see that Yioop transparently supports any language whose Unicode letters can be provided as input. Section 7 discusses how the query history of a user can be used to come up with better suggestions. This section concludes the discussion on autosuggest in Yioop. Section 8 goes beyond autosuggestion to discuss spelling correction for queries. We will discuss the efficacy of the Edit-Distance algorithm [3]. Section 9 tackles the problem of rendering suggestions for non-English queries transliterated into English. The South-Indian language Telugu [4] will be used as an example. Finally, we will discuss the performance of the system and future work. A list of references has been included at the end.

2. YIOOP SEARCH ENGINE

Yioop is a PHP-based search engine designed to allow users to produce indexes of a website or a collection of websites [5]. Currently, Yioop does not support any form of autosuggestion or spelling correction when a user types in a query. Our goal will be to add these features to Yioop.

Javascript will be used to develop the functionality in the front-end so that the Yioop servers will not be affected. It was a challenge to develop these computationally intensive tasks using Javascript and some trade-offs had to be made that will be discussed when we talk about performance.

3. PRELIMINARY WORK

Some of the different kinds of applications in which autosuggest has been implemented, are listed below [2]. This list serves to provide a context for how this feature will be modified for Yioop.

Web Browsers – Here autosuggest is generally used in the address bar to suggest relevant URLs (Uniform Resource Locators) while the user is typing.

Search Engines – Here autosuggest is used to suggest relevant queries, even corrections, to the user while typing. Commercial search engines provide useful suggestions based on earlier user behavior aggregated across logs. Apart from just the text being typed, additional signals such as a user's search history, the geographic location and the currently trending search queries can also be considered. Understanding more signals and context results in a better search experience. Google Instant is a very popular implementation of such autosuggest techniques [2].

Word-processors - In many word processing programs, auto-completion decreases the amount of time spent typing repetitive words and phrases. The corpus used for auto-completion is usually not just a static dictionary, but also a dynamic one that can be extended with the text in the document itself and also any user-specified list of words [2].

Code Editors – Popular IDEs (Interactive Development Environments) like Eclipse use autosuggestion techniques to help the user type in long programs. It saves time and avoids spelling errors, which in coding environments can mean a broken program. For example, using auto-complete for long variable names like 'numberOfLinesOfCode' can reduce errors and burden. The user need not worry anymore of longer and more descriptive names, which in turn improves the quality of code [2].

3.1 DICTIONARY WORDS

A basic necessity for developing autosuggest in any language is a set of relevant and frequently used words in that language. The quality of the feature depends upon how comprehensive the corpus is and how sophisticated the front-end design is. It is possible to work with a compact list of dictionary root words but we will require very detailed stemming logic in the front-end. Similarly, a straightforward front-end can work if the supporting corpus contains most of the variations of the root word. For our project, the corpus has been chosen from Wiki sources [6].

3.2 STORING DICTIONARY WORDS

Data structures like tries, suffix trees and ternary search trees were considered to store dictionary words for the implementation. After researching their operational time and space complexities, it was judged that a trie would be most appropriate. [7] Figure 3.1 depicts the structure of a trie:

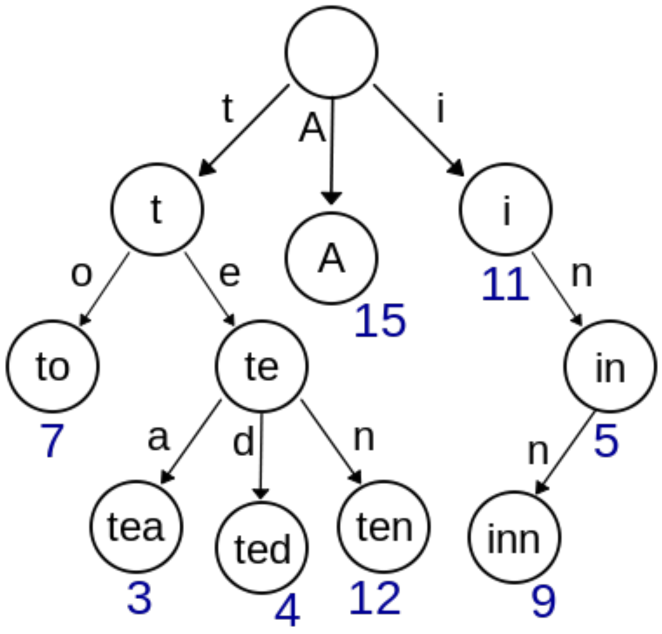


Figure 3.1 Example of Trie [7]

The figure represents a trie constructed with English words. Every arrow represents a letter in the alphabet and every node represents the word formed by a sequence of arrows from the root that terminates at that node.

Programmatically this was achieved by using multi-level arrays in PHP. The initial version of code for constructing a trie did the following -

- Created a trie in which words are stored using multi-level PHP arrays.
- The trie was then JSON encoded and a gzip version was created.
- Any words with less than 3 letters or stop words [8] or any words which has non-ASCII characters
- The final gzip file was around 250KB, which was a reasonable size to send over the network and load while Yioop website is loaded.

Figure 3.2 gives the structure of the actual constructed trie.

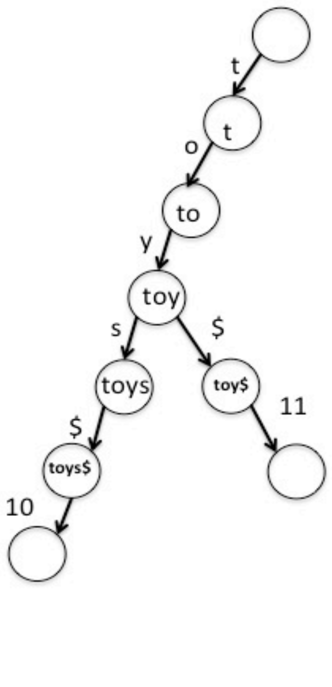


Figure 3.2 Structure of generated trie

Every complete word uses the delimiter ‘$’, followed by its frequency in our input corpus. The frequency is used to pick the most relevant choice out of many possible outcomes when doing spelling correction.

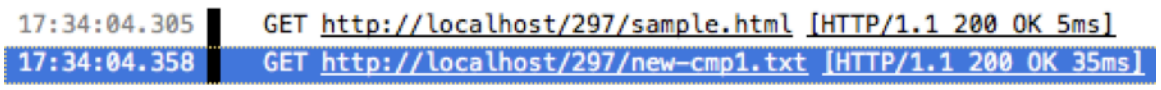
The constructed trie is stored on the Yioop server and delivered whenever a user accesses the Yioop website. The downloaded trie is further processed using Javascript on the client-side

3.3 TIMING TESTS

Timing tests were conducted to see how the addition of trie effects the page load time of Yioop website. Time was monitored using Firefox Web Console option [9]. This helps in finding load time profiles for a given website.

With the aim of achieving the lowest possible load time, the following options were considered.

1. Loading a plain JSON-encoded trie of size 2.5MB took around 2.5 seconds.
2. When the gzip option of HTTP is enabled, the trie can be compressed and decompressed on the fly. Even with the additional computation this entails, the network latency was lower due to the smaller size of the file being sent and the response time was around 400 ms, much faster than loading a trie directly.
3. The next option was to send the compressed trie in a text file, rather than having HTTP doing it for every request. This reduces load on the server because it no longer has to compress the 2.5 MB trie on the fly but the client still has to decompress the file. Downloading the zipped file , with a size of about 250 KB, followed by client-side decompression took around 35ms as shown in Figure 3.3. This figure shows the load times as displayed in the Firefox Web Console.

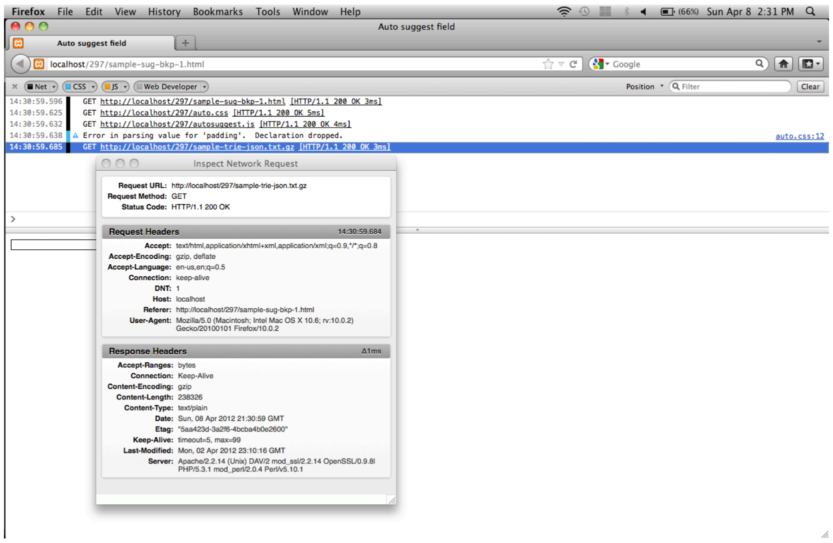


```
17:34:04.305    GET http://localhost/297/sample.html [HTTP/1.1 200 OK 5ms]
17:34:04.358    GET http://localhost/297/new-cmp1.txt [HTTP/1.1 200 OK 35ms]
```

Figure 3.3: Load time of compressed trie with sent in a text file

4. The final experiment was to compress the trie with a .gz extension while enabling the HTTP option ‘gzip,deflate’. When this option is provided, the browser expects a compressed file and decompresses it on the fly. With this approach, it takes just 3 ms

for the browser to download the compressed trie, decompress it and make the JSON-encoded trie available for autosuggest (See Figure 3.4)



```
14:30:59.596   GET http://localhost/297/sample-sug-bkp-1.html [HTTP/1.1 200 OK 3ms]
14:30:59.625   GET http://localhost/297/auto.css [HTTP/1.1 200 OK 5ms]
14:30:59.632   GET http://localhost/297/autosuggest.js [HTTP/1.1 200 OK 4ms]
14:30:59.638   Error in parsing value for 'padding'.  Declaration dropped.        auto.css:12
14:30:59.685   GET http://localhost/297/sample-trie-json.txt.gz [HTTP/1.1 200 OK 3ms]
```

```
    Request URL: http://localhost/297/sample-trie-json.txt.gz
 Request Method: GET
    Status Code: HTTP/1.1 200 OK

Request Headers                                               14:30:59.684
           Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  Accept-Encoding: gzip, deflate
  Accept-Language: en-us,en;q=0.5
       Connection: keep-alive
              DNT: 1
             Host: localhost
          Referer: http://localhost/297/sample-sug-bkp-1.html
       User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:10.0.2)
                   Gecko/20100101 Firefox/10.0.2

Response Headers                                                      Δ1ms
    Accept-Ranges: bytes
       Connection: Keep-Alive
 Content-Encoding: gzip
   Content-Length: 238326
     Content-Type: text/plain
             Date: Sun, 08 Apr 2012 21:30:59 GMT
             Etag: "5aa423d-3a2f6-4bcba4b0e2600"
       Keep-Alive: timeout=5, max=99
    Last-Modified: Mon, 02 Apr 2012 23:10:16 GMT
           Server: Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l
                   PHP/5.3.1 mod_perl/2.0.4 Perl/v5.10.1
```

Figure 3.4: Load time of .gz trie with browser enabled to accept gzip files

The experiments are summarized in the Table 3.1

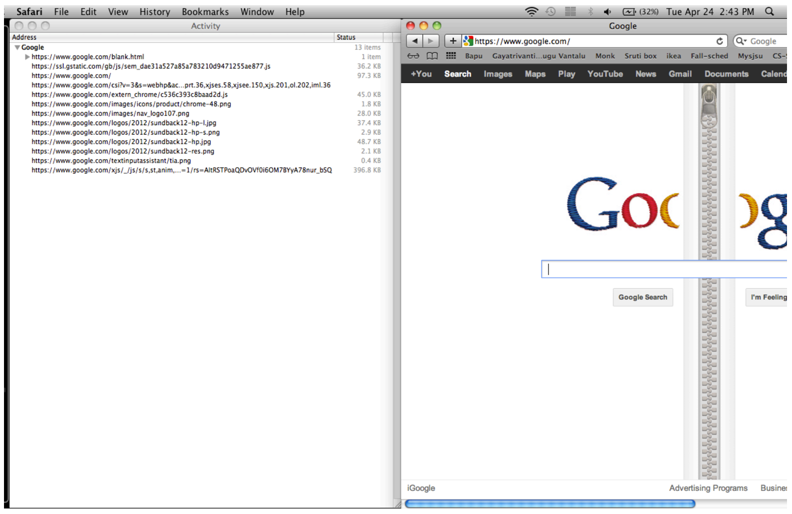
Trie type	Size in KB	Response Time in ms
Plain JSON	*2500*	*2500*
Plain JSON with gzip enabled on HTTP	*2500*	*400*
File with compressed JSON data	*250*	*35*
Zipped JSON file with deflate option	*110*	*3*

Table 3.1 Comparison of load times of different formats of tries

After conducting these experiments, a compressed trie with a .gz extension was made available on Yioop server that will be downloaded when the page is loaded with a browser enabled to accept gzip files.

3.4 GOOGLE AUTOSUGGEST

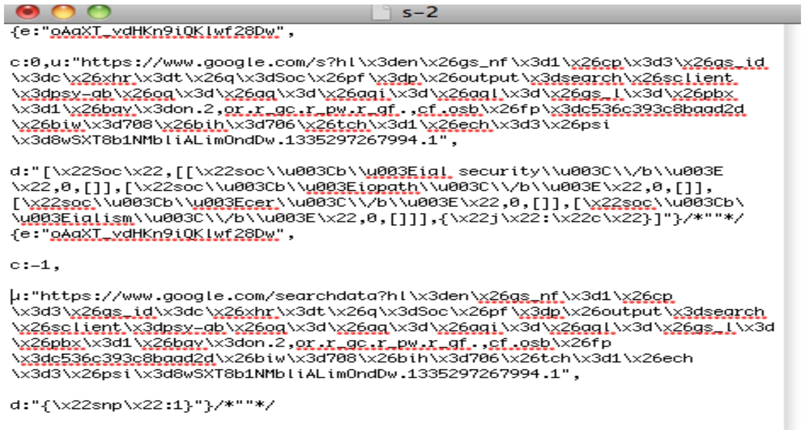
An experiment was conducted on how Google’s autosuggest functionality works. The aim was to reverse engineer the process in order to help develop a better suggestion feature for Yioop. The Activity Window in the Safari web browser was used as the tool. A series of experiments were conducted on Google search page with various kinds of inputs. The actions going on in the back end were then examined using the Safari Activity window. The activity window display on opening a Google search page looks some thing like Figure 3.5.



```
Address                                                                              Status
▼ Google                                                                           13 items
  ▶ https://www.google.com/blank.html                                                1 item
    https://ssl.gstatic.com/gb/js/sem_dae31a527a85a783210d9471255ae877.js          36.2 KB
    https://www.google.com/                                                        97.3 KB
    https://www.google.com/csi?v=3&s=webhp&ac...prt.36,xjses.58,xjsee.150,xjs.201,ol.202,iml.36
    https://www.google.com/extern_chrome/c536c393c8baad2d.js                       45.0 KB
    https://www.google.com/images/icons/product/chrome-48.png                       1.8 KB
    https://www.google.com/images/nav_logo107.png                                  28.0 KB
    https://www.google.com/logos/2012/sundback12-hp-l.jpg                          37.4 KB
    https://www.google.com/logos/2012/sundback12-hp-s.png                           2.9 KB
    https://www.google.com/logos/2012/sundback12-hp.jpg                            48.7 KB
    https://www.google.com/logos/2012/sundback12-res.png                            2.1 KB
    https://www.google.com/textinputassistant/tia.png                               0.4 KB
    https://www.google.com/xjs/_/js/s/s,st,anim,...=1/rs=AItRSTPoaQDvOVf0i6OM7BYyA78nur_bSQ   396.8 KB
```

Figure 3.5: Google search page being inspected in Safari activity window

The data loaded after every click can be inspected using the activity window. The downloaded file contains JSON data with the URL in hex format and the URL contains all the suggested values for the query. The file downloaded for the search query "Soc" can be seen in Figure 3.6.



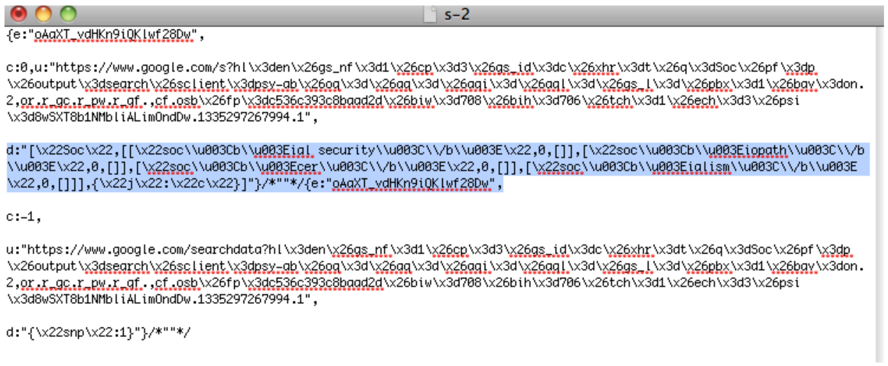
```
s-2
{e:"oAqXT_vdHKn9iQKlwf28Dw",

c:0,u:"https://www.google.com/s?hl\x3den\x26gs_nf\x3d1\x26cp\x3d3\x26gs_id
\x3dc\x26xhr\x3dt\x26q\x3dSoc\x26pf\x3dp\x26output\x3dsearch\x26sclient
\x3dpsy-ab\x26oq\x3d\x26aq\x3d\x26aqi\x3d\x26aql\x3d\x26gs_l\x3d\x26pbx
\x3d1\x26bav\x3don.2,or.r_gc.r_pw.r_gf.,cf.osb\x26fp\x3dc536c393c8baad2d
\x26biw\x3d708\x26bih\x3d706\x26tch\x3d1\x26ech\x3d3\x26psi
\x3d8wSXT8b1NMbliALimOndDw.1335297267994.1",

d:"[\x22Soc\x22,[[\x22soc\\u003Cb\\u003Eial security\\u003C\\/b\\u003E
\x22,0,[]],[\x22soc\\u003Cb\\u003Eiopath\\u003C\\/b\\u003E\x22,0,[]],
[\x22soc\\u003Cb\\u003Ecer\\u003C\\/b\\u003E\x22,0,[]],[\x22soc\\u003Cb\
\u003Eialism\\u003C\\/b\\u003E\x22,0,[]]],{\x22j\x22:\x22c\x22}]"}/*""*/
{e:"oAqXT_vdHKn9iQKlwf28Dw",

c:-1,

u:"https://www.google.com/searchdata?hl\x3den\x26gs_nf\x3d1\x26cp
\x3d3\x26gs_id\x3dc\x26xhr\x3dt\x26q\x3dSoc\x26pf\x3dp\x26output\x3dsearch
\x26sclient\x3dpsy-ab\x26oq\x3d\x26aq\x3d\x26aqi\x3d\x26aql\x3d\x26gs_l\x3d
\x26pbx\x3d1\x26bav\x3don.2,or.r_gc.r_pw.r_gf.,cf.osb\x26fp
\x3dc536c393c8baad2d\x26biw\x3d708\x26bih\x3d706\x26tch\x3d1\x26ech
\x3d3\x26psi\x3d8wSXT8b1NMbliALimOndDw.1335297267994.1",

d:"{\x22snp\x22:1}"}/*""*/
```

Figure 3.6: Google search page being inspected in Safari activity window for query "Soc"

In the process of analyzing the exchange of data between the browser and the Google server, it was found that URLs have the following characteristics:

- 'nf' is always 1
- 'cp' numbers the urls 1,2,3...
- 'gs_id' is the Google search Id
- 'tch' is always 1
- 'ech' is an index like 'cp', but it will be the same if a same query is entered twice during one search

Figure 3.7 below shows the highlighted part where we can see the words that are retrieved for the autosuggest list



```
s-2
{e:"oAqXT_vdHKn9iQKlwf28Dw",

c:0,u:"https://www.google.com/s?hl\x3den\x26gs_nf\x3d1\x26cp\x3d3\x26gs_id\x3dc\x26xhr\x3dt\x26q\x3dSoc\x26pf\x3dp
\x26output\x3dsearch\x26sclient\x3dpsy-ab\x26oq\x3d\x26aq\x3d\x26aqi\x3d\x26aql\x3d\x26gs_l\x3d\x26pbx\x3d1\x26bav\x3don.
2,or.r_gc.r_pw.r_gf.,cf.osb\x26fp\x3dc536c393c8baad2d\x26biw\x3d708\x26bih\x3d706\x26tch\x3d1\x26ech\x3d3\x26psi
\x3d8wSXT8b1NMbliALimOndDw.1335297267994.1",

d:"[\x22Soc\x22,[[\x22soc\\u003Cb\\u003Eial security\\u003C\\/b\\u003E\x22,0,[]],[\x22soc\\u003Cb\\u003Eiopath\\u003C\\/b
\\u003E\x22,0,[]],[\x22soc\\u003Cb\\u003Ecer\\u003C\\/b\\u003E\x22,0,[]],[\x22soc\\u003Cb\\u003Eialism\\u003C\\/b\\u003E
\x22,0,[]]],{\x22j\x22:\x22c\x22}]"}/*""*/{e:"oAqXT_vdHKn9iQKlwf28Dw",

c:-1,

u:"https://www.google.com/searchdata?hl\x3den\x26gs_nf\x3d1\x26cp\x3d3\x26gs_id\x3dc\x26xhr\x3dt\x26q\x3dSoc\x26pf\x3dp
\x26output\x3dsearch\x26sclient\x3dpsy-ab\x26oq\x3d\x26aq\x3d\x26aqi\x3d\x26aql\x3d\x26gs_l\x3d\x26pbx\x3d1\x26bav\x3don.
2,or.r_gc.r_pw.r_gf.,cf.osb\x26fp\x3dc536c393c8baad2d\x26biw\x3d708\x26bih\x3d706\x26tch\x3d1\x26ech\x3d3\x26psi
\x3d8wSXT8b1NMbliALimOndDw.1335297267994.1",

d:"{\x22snp\x22:1}"}/*""*/
```

Figure 3.7: Downloaded file with data on autosuggest results for "Soc"

As we continue to type the query, the results will be displayed using the topmost suggested word. A similar file to the above can be downloaded that has the styling and content of the result page. It can be seen that every request is being sent to the server and some kind of relevant data is being fetched. Reverse engineering the complete autosuggest process was not possible, but this experiment gave a good understanding of the sequence of events that occur during a search.

As mentioned earlier, it was decided that autosuggest in Yioop will to be implemented on the front end. Hence, following such a method is not possible. Yioop needs to make use of only the local content.

4. AUTOSUGGESTION IN YIOOP

Incorporating the basic autosuggest functionality in Yioop was the next step. The initial corpus was a list of English dictionary words [2]. Javascript code was used to infer the relevant words for autosuggestion on the client-side. The steps are as follows:

- The trie is downloaded when the Yioop page is loaded.
- A Javascript handler for the 'onKeyUp' event delivers the typed letters to the program. As the letters accumulate, relevant suggestion words are retrieved and displayed.
- Only the top six words are made visible but the user can scroll down to see more results.
- The user can hover the cursor on the suggestions and click one of them as the query.
- Otherwise, the user can also use the arrow keys to traverse through the list and press the Enter key to submit the query

A screenshot of Yioop autosuggest in use is displayed in the Figure 4.1. This image shows the dropdown for single character entry 'c'.

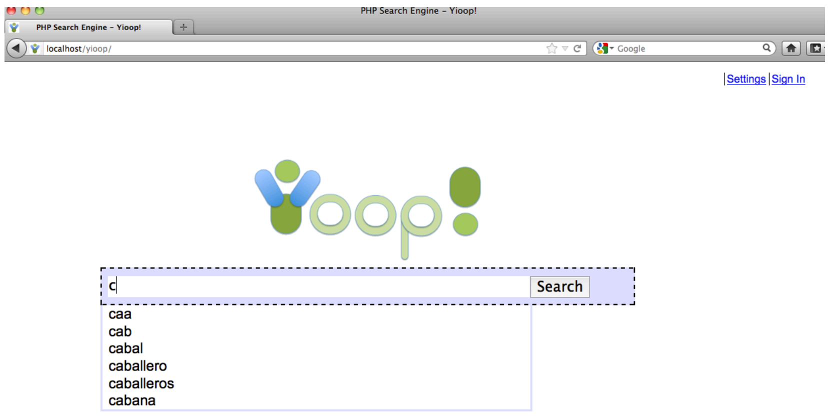
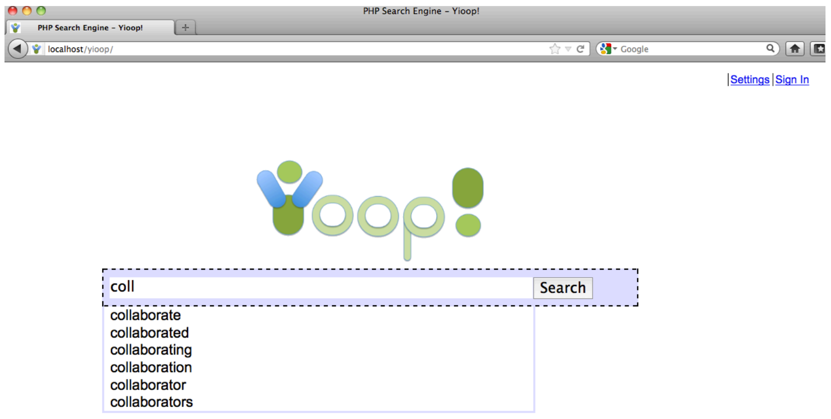


Figure 4.1: Yioop suggest for character 'c'

This is the initial version for the layout for the suggestion list. It has evolved with the project and is evident in the further figures. For a search term of more than one character, it looks something like Figure 4.2.

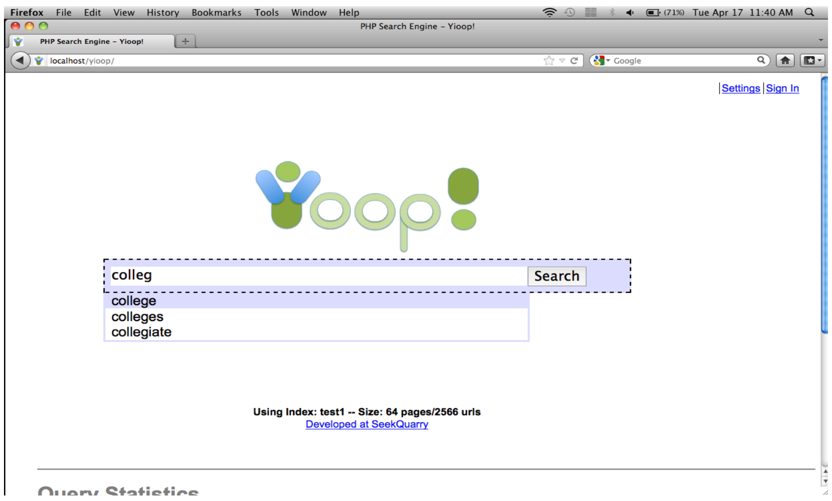


coll

collaborate
collaborated
collaborating
collaboration
collaborator
collaborators

Figure 4.2: Yioop suggest for word ‘coll’

After the intended word is found in the dropdown, the user can click it to select it. See Figure 4.3.



colleg

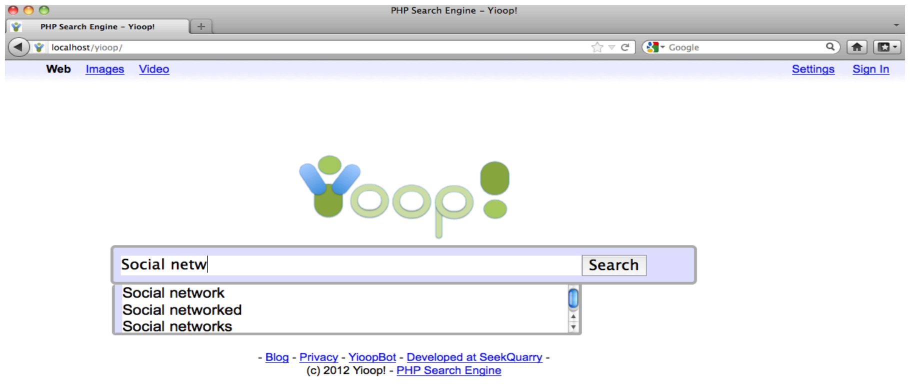
college
colleges
collegiate

Using Index: test1 -- Size: 64 pages/2566 urls

Figure 4.3: User selecting a word from autosuggest list

5. MULTI-WORD SUGGEST

After developing the basic autosuggest feature, more features were added to make the suggestions more useful. Multi-word suggestions were introduced. Figure 5.1 shows an example of multi-word suggestion. Whenever a space is typed in a query, the suggest list prepends the previous words of the query to the later suggestions. This way, possible phrases are suggested.

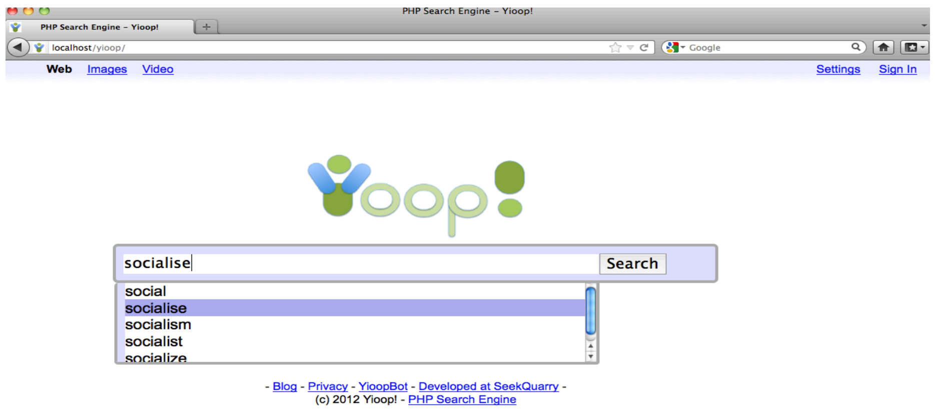

PHP Search Engine – Yioop!
localhost/yioop/
Web
Images
Video
Settings
Sign In
Search
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Social netw

Social network
Social networked
Social networks

Figure 5.1: Multi-word suggestion for query "Social netw"

Figure 5.2 shows that using the arrow keys, a word from the suggestion list can be selected and placed in the search box and the intended query can be submitted using the Enter key. A new layout for the suggestion list can be seen in the figure.


PHP Search Engine – Yioop!
localhost/yioop/
Web
Images
Video
Settings
Sign In
Search
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

socialise

social
socialise
socialism
socialist
socialize

Figure 5.3: Usage of up/down arrow keys on the list

6. FOREIGN INPUT SUPPORT

With the growing popularity of search engines in every part of the world, it is very important to be able to support different sets of input for the suggestions. Yioop has the flexibility to add support for additional languages, and autosuggest feature was likewise updated to work for any foreign input.

Internationalizing the trie construction and suggestion technique was accomplished by handling the Unicode representations of any give character. Currently this has been tested for French and Russian, but it will support any language whose characters are present in the Unicode character set.

Below are snapshots for French word suggestions.

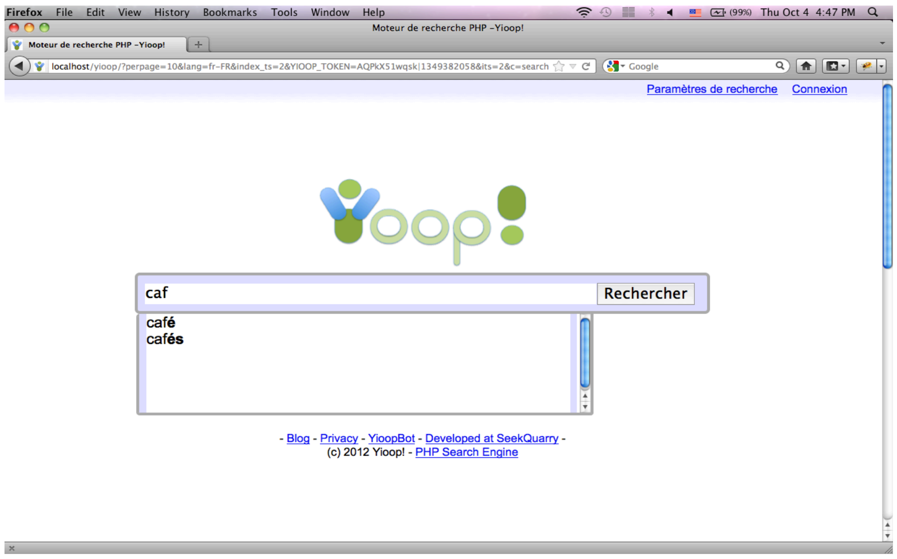

Firefox File Edit View History Bookmarks Tools Window Help
Thu Oct 4 4:47 PM
Moteur de recherche PHP –Yioop!
localhost/yioop/?perpage=10&lang=fr-FR&index_ts=2&YIOOP_TOKEN=AQPkX51wqsk|1349382058&its=2&c=search
Paramètres de recherche
Connexion
caf
Rechercher
café
cafés
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 6.1 French word suggestions

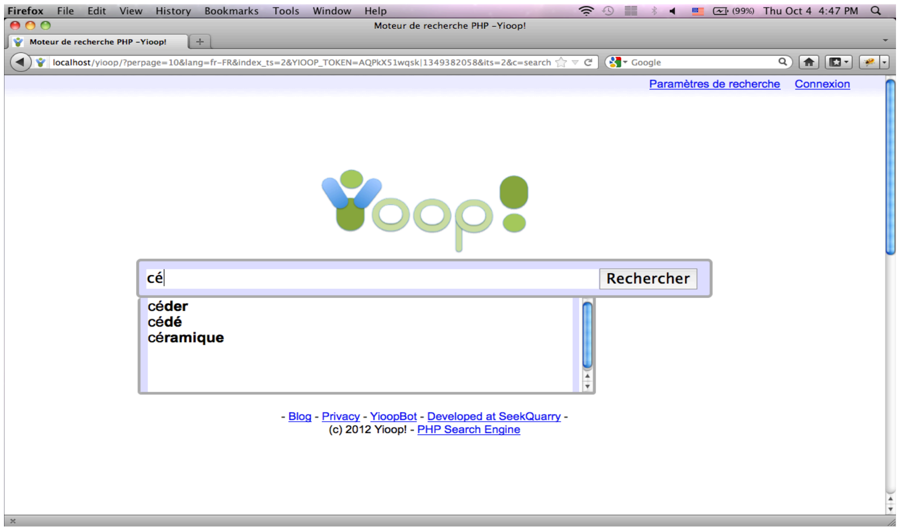

Moteur de recherche PHP –Yioop!
localhost/yioop/?perpage=10&lang=fr-FR&index_ts=2&YIOOP_TOKEN=AQPkX51wqsk|1349382058&its=2&c=search
Paramètres de recherche
Connexion
cé
Rechercher
céder
cédé
céramique
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 6.2 French word suggestions

Below are the snapshots for Russian word suggestions.

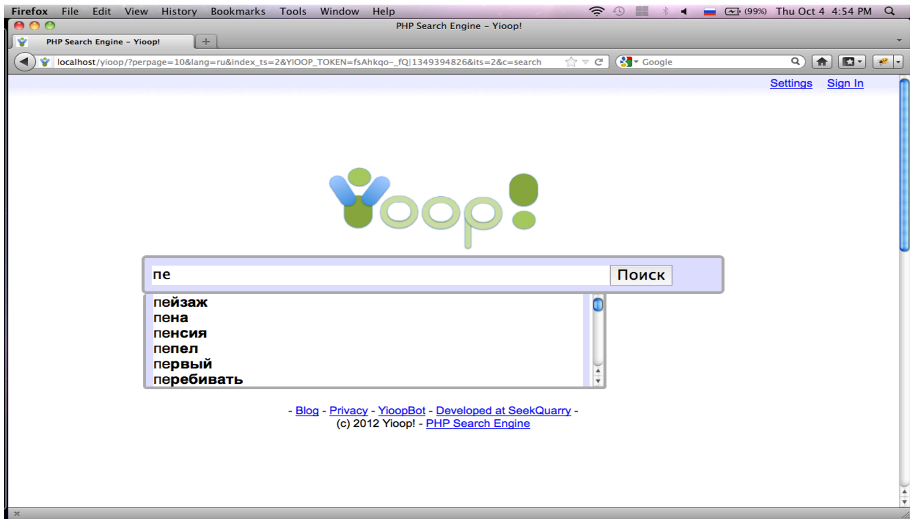

PHP Search Engine – Yioop!
localhost/yioop/?perpage=10&lang=ru&index_ts=2&YIOOP_TOKEN=fsAhkqo-_fQ|1349394826&its=2&c=search
Settings
Sign In
пе
Поиск
пейзаж
пена
пенсия
пепел
первый
перебивать
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 6.3 Russian word suggestions

7. USING LOCAL STOARGE TO SUGGESTIONS

Now that the simple suggestion of words from the dictionary is supported, this section tries to enhance the autosuggestion by storing the previous queries in the web browser's local storage. With HTML5, the user's browser is capable of storing data locally. This type of storage proved to faster and secure as the data is not included only when it is asked for but not every server request. Large amounts of data can be stored in key/value pairs without affecting the page's performance and a web page can only access data that it stored [10].

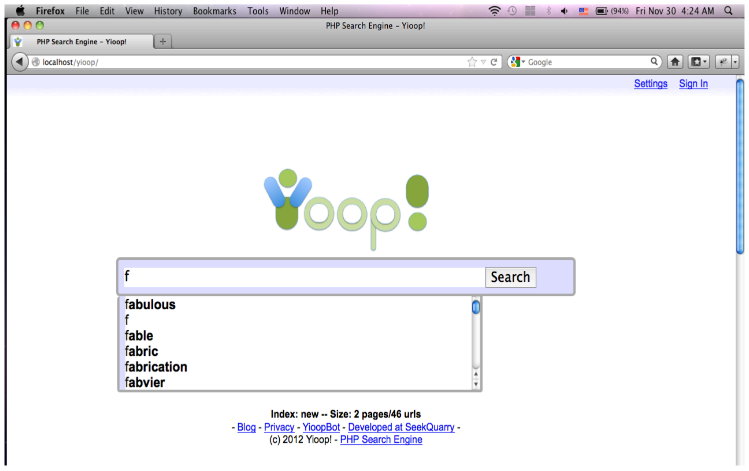
7.1 ALGORITHM FOR SUGGESTIONS USING LOCAL STORAGE

Whenever a user presses Enter to submit a query, the query in the search box will be placed in local storage for future use. The following steps are taken:

1. When the user presses Enter the query is stored. The storage is specific to each 'locale' used by Yioop. For example, the locale for US English is 'en-US'. Storing in such a way accommodates all existing languages supported by Yioop and leaves room for new languages. Also, a trie of all the local storage query words is constructed and stored for future use. For example, when a user is in English language mode, and submits a query 'fabric', it is stored as follows
   en-US_0 -> {"f":{"a":{"b":{"r":{"i":{"c":{"$":"$"}}}}}}}@@{"fabric":1}
   - en-US is the locale specific to US English.
   - 0 is the version number for the Javascript file that has the implementation for the suggestions. This is manually changed when the file undergoes changes. This is checked every time a user has suggestions in local storage, and if the version number does not match then the local storage will be deleted. This helps to eliminate any stale data.
   - The query 'fabric' is stored as a trie in JSON format
   - It is followed by the actual query along with the number of times that query is fired and '@@' is a unique delimiter

2. When a user types a query the next time, first the local storage is checked for any existing suggestions.
3. If available, they appear first in the suggest list and are listed in descending order by the total number of times they have been fired.
4. The actual dictionary is searched for further suggestions.

As Yioop does not have any user base query data to rely on for suggestions, the best alternative is to store the user's previous queries to reduce the typing work. Figure 7.1 shows that the term 'fabrication' is first in the suggest list as it has been quereid before


Firefox File Edit View History Bookmarks Tools Window Help
Fri Nov 30 4:24 AM
PHP Search Engine - Yioop!
localhost/yioop/
Google
Settings
Sign In
Search

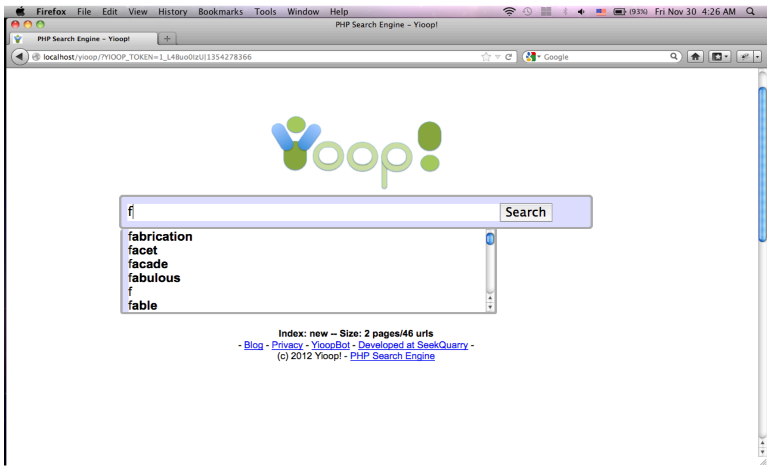
f

fabulous
f
fable
fabric
fabrication
fabvier

Index: new -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 7.1 Local storage suggestions

Figure 7.2 shows the suggestions from local storage ordered by their frequencies.


Firefox File Edit View History Bookmarks Tools Window Help
Fri Nov 30 4:26 AM
PHP Search Engine - Yioop!
localhost/yioop/?YIOOP_TOKEN=1_L4Buo0IzU|1354278366
Google
Search

f

fabrication
facet
facade
fabulous
f
fable

Index: new -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

7.2 Local storage suggestions ordered by frequency

8. SPELLING CORRECTION

It is quite possible that a user might spell words incorrectly while entering queries in Yioop. This section deals with the spell correction technique used to suggest corrections for misspelled words in a query. Many commercial search-engines like Google use this feature and ask users ‘Did you mean:’ with the most relevant spelling correction. Using popular search data to will also help in giving the most relevant spell corrections.

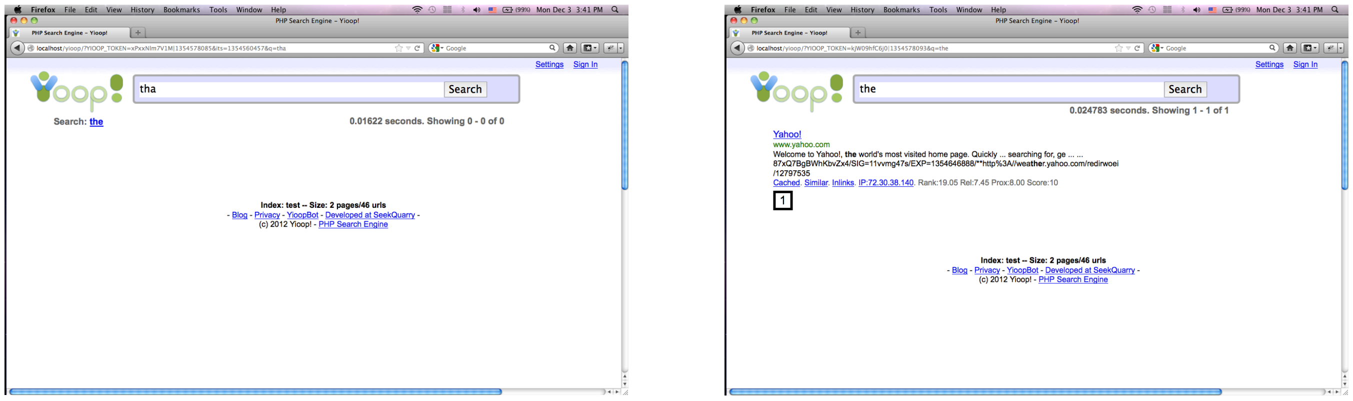
In the constrained environment of Yioop where there is no external data, a well-picked dictionary with a set of frequent words is used for spelling correction. This is the same dictionary used for autocomplete suggestions. Spelling correction may involve usage of different kinds of data similar to suggestions.

8.1 ALGORTHM FOR SPELL CORRECTION

The most intuitive algorithm for correcting typing errors is the Edit Distance algorithm [3]. The number of edits it would take to turn into correct word is the edit distance between the two words. There can be different kinds of edits between two given words. It can be a deletion where a letter is removed, a transposition where there is a swap of adjacent letters, a replacement where another replaces a letter or an insertion where an unwanted letter is inserted [11].

As discussed in [11], for a word of length n, there will be n deletions, n-1 transpositions, 26n alterations, and 26(n+1) insertions, for a total of 54n+25 (of which a few are typically duplicates). For example, for the word ‘improve’ the candidates with edit distance – 1 will be is 390. The dictionary words are selected from this candidate list and sorted by their frequency of usage. The algorithm is as follows:

```
SpellCorrection (word):
        Candidates = known (word) or known (Edits1 (word)) or word
        Return candidate with maximum frequency in the trie
Edits1 (word):
        Deletes: Set of words with one letter deleted
        Transposes: Set of words with a swap between the adjacent characters
        Replaces: Set of words with every letter replaced by 25 other letters in
        English alphabet
        Inserts: Addition of an unwanted letter at all given positions in a word
        Known (words):
        Returns the set of words that are in the dictionary
```


Firefox File Edit View History Bookmarks Tools Window Help
Mon Dec 3 3:41 PM
PHP Search Engine – Yioop!
localhost/yioop/?YIOOP_TOKEN=xPxxNlm7V1M|1354578085&its=1354560457&q=tha
Google
Settings Sign In
tha
Search

Search: the

0.01622 seconds. Showing 0 - 0 of 0

Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Firefox File Edit View History Bookmarks Tools Window Help
Mon Dec 3 3:41 PM
PHP Search Engine – Yioop!
localhost/yioop/?YIOOP_TOKEN=kjW09hfC6j0|1354578093&q=the
Google
Settings Sign In
the
Search

0.024783 seconds. Showing 1 - 1 of 1

Yahoo!
www.yahoo.com
Welcome to Yahoo!, **the** world's most visited home page. Quickly ... searching for, ge ... ...
87xQ7BgBWhKbvZx4/SIG=11vvmg47s/EXP=1354646888/**http%3A//wea**the**r.yahoo.com/redirwoei
/12797535
Cached. Similar. Inlinks. IP:72.30.38.140. Rank:19.05 Rel:7.45 Prox:8.00 Score:10

1

Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 8.1 Correction of incorrect query 'tha'

It can be seen in Figure 8.1 that when an incorrect query 'tha' is submitted, the text 'Search:' along with the corrected word 'the' appears on the results page. By clicking the corrected word, search results appear as it can be seen in the right image.

8.2 EVALUATION

About 90% of spelling errors are of edit distance 1 as claimed by the literature on spelling correction [11], but it is also quite possible that the spelling errors are an edit distance of 2. Edit distance 2 is nothing there is a possibility of 2 insertion, deletions, replacements or transpositions. Using the above algorithm, this can be achieved by just applying the edit distance 1 algorithm to the set of words that were the output from applying the initial edit distance algorithm. This involves huge computation. The number of candidates when edit distance 2 is applied to the word 'improve' is 1,62,150.

As this feature is being coded on the front-end using Javascript, such a huge computation is undesirable. Small experiment was conducted on how edit distances 1 and 2 perform for a given set of words.

Query	*Corrections*	
	Edit Distance 1	*Edit Distance 2*
tha	*the*	*the*
lagh	*laugh*	*last*
sceince	*science*	*since*
nees	*news*	*been*
latre	*later*	*are*

Table 8.1 Experiment on spelling corrections using Edit distance 1 and 2

A set of words ranging from most frequently used words to words that are more prone to spelling errors have been used for the experiment. Table 8.1 depicts the sample set. Considering the case where the rate of spelling errors with edit distance 1 is more, it is evident from the experiment that, edit distance 1 gives better results than edit distance 2. Hence, a tradeoff should be made in order to lower the burden of computation for Yioop. It has been concluded that using just the edit distance 1 would serve the purpose in most of the cases and also will avoid the computational overhead of edit distance 2 algorithm.

9. SUGGESTIONS FOR TRANSLITERATED QUERIES

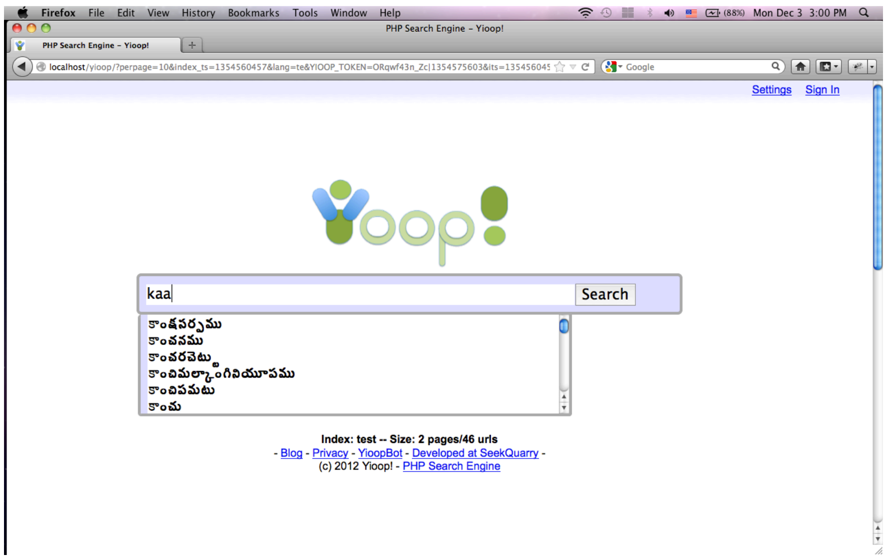
Transliteration is the process of mapping text written in one language in to another by means of a pre-defined mapping [13]. With the rapid growth of Internet usage via emails, social networking sites, blogs etc. it has been common to use English transliteration for foreign languages. Users who do not know that script of a particular language, tend to use this method. In the case of unavailability of a direct method to input data in a given language, transliteration becomes handy. Hence, transliteration can be understood as the process of entering data in one language using the script of another language [13].

To experiment with this feature, Telugu, the third most spoken language in India, has been chosen. Telugu has 56 alphabets (18 vowels and 38 consonants), two of which are not in use now. There are 16 additional Unicode characters to represent the phonetic variants of each consonant [13].

Every phoneme in Telugu script when transliterated using English ends with a vowel. Based on this, the approach of constructing a mapping table has been chosen. A key-value data structure has been constructed to accommodate most transliterations for the single characters. Below is the sample data to show how the data structure is constructed.

```
telugu_array['k']='క్';                              telugu_array['+aa']=ా';

telugu_array['kh']='ఖ్';                             telugu_array['+oo']=ో
```

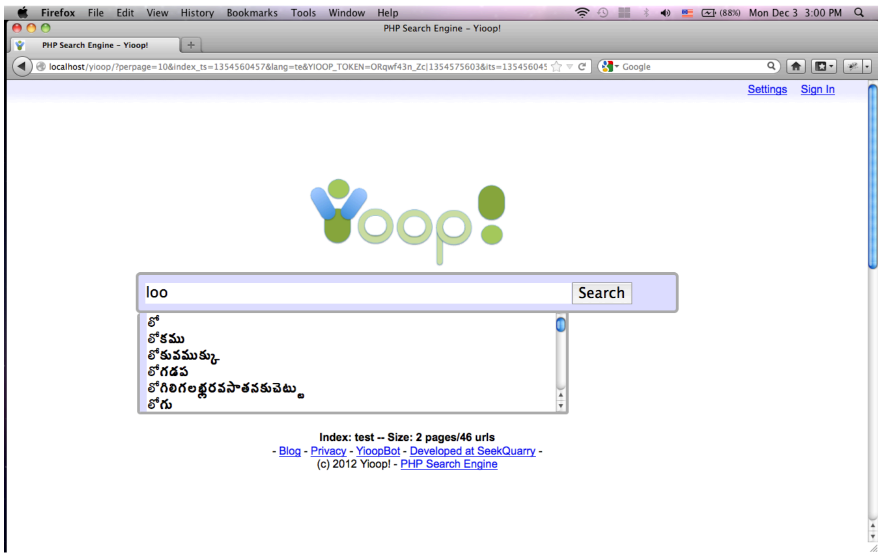
The algorithm used to transliterate divides the input query into chunks, such as a set of 2 or 3 characters, ending in a vowel and compare them against the table of entries. After analyzing various valid combinations, input query was modified to Telugu script. Using that modified query, suggestions are listed using the Telugu dictionary trie [14]. Figure 9.1 and 9.2 show two examples.


Firefox File Edit View History Bookmarks Tools Window Help
Mon Dec 3 3:00 PM
PHP Search Engine – Yioop!
localhost/yioop/?perpage=10&index_ts=1354560457&lang=te&YIOOP_TOKEN=ORqwf43n_Zc|1354575603&its=13545604
Google
Settings
Sign In
Search
kaa

కాంక్షసర్పము
కాంచనము
కాంచరచెట్టు
కాంచిమల్కాంగినియూపము
కాంచిపమటు
కాంచు

Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 9.1 Suggestions for Telugu term 'కా' transliterated as 'kaa' in English


Firefox File Edit View History Bookmarks Tools Window Help
Mon Dec 3 3:00 PM
PHP Search Engine – Yioop!
localhost/yioop/?perpage=10&index_ts=1354560457&lang=te&YIOOP_TOKEN=ORqwf43n_Zc|1354575603&its=13545604
Google
Settings
Sign In
Search
loo

లో
లోకము
లోకువముక్కు
లోగడప
లోగిలిగలల్లరవసాతనకుచెట్టు
లోగు

Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 9.2 Suggestions for Telugu term 'లో' transliterated as 'loo' in English

10. PERFORMANCE

The performance of these features can be tested in different ways. Following were the experiments conducted.

Experiment 1: Queries were typed in the Yioop search box in following two modes.

- ‘Word Suggest’ option disabled
- ‘Word Suggest’ option enabled

Following are the times recorded for five different people with different typing speeds.

Word	Without Autosuggest – Time in sec					With Autosuggest – Time in sec				
	Per 1	Per 2	Per 3	Per 4	Per 5	Per 1	Per 2	Per 3	Per 4	Per 5
Science	4	5	4	4.5	3	2	3	2	3.5	2
Computer	5	4	4.5	4.5	3	3	2.5	3	3	2
Adapt	3.4	3.5	3	3	2	3	3	2	2	1.5
Accomplish	5	4	4.5	4.5	3	3	2.5	3	3	2

Table 10.1 Comparison of a user’s typing time with and without autosuggest

It is evident that using autosuggestion considerably reduces the typing work of a user.

Experiment 2: To compare autosuggest against browser’s autocomplete feature :

Even this experiment involves two modes as mentioned above. To conduct this experiment, the following has been done.

Word suggest is turned off and the queries ‘screen’, ‘scare’ and ‘science’ have been searched 5 , 3 and 3 times respectively. Later, the same searches have been done in Yioop with the ‘Word Suggest’ option on. The following are the outcomes.

Multi-word suggest:

Figure 10.1 depicts the outcome when in two modes, for the query ‘screen s’

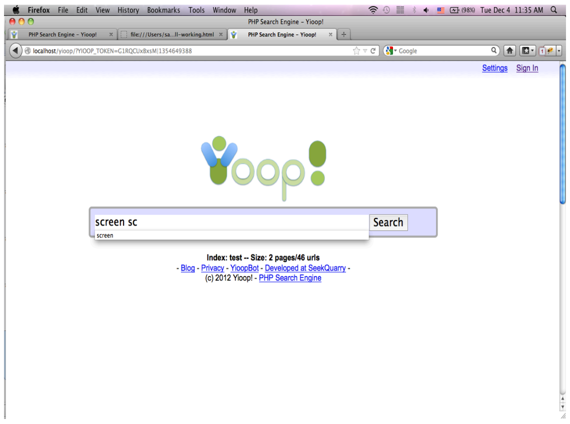
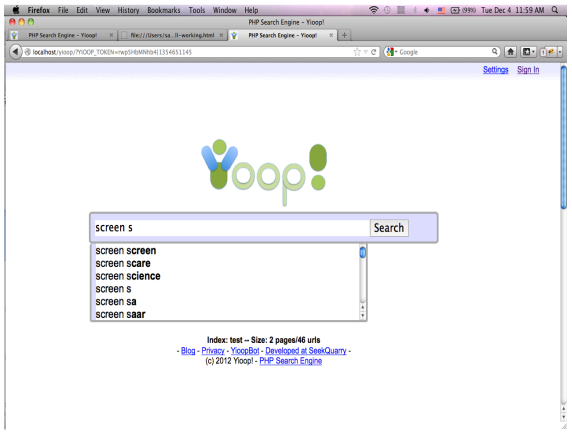

Settings
Sign In
Yioop!
screen sc
Search
screen
Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Settings
Sign In
Yioop!
screen s
Search
screen screen
screen scare
screen science
screen s
screen sa
screen saar
Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 10.1 Comparison of multi-word suggest

It can be seen that, with browser autocompletion, once the second word is typed there are no valid suggestions even from the stored list of previous queries. But, autosuggest appends the previous query terms and renders phrases so that user can choose from the list.

Usage of previous queries :

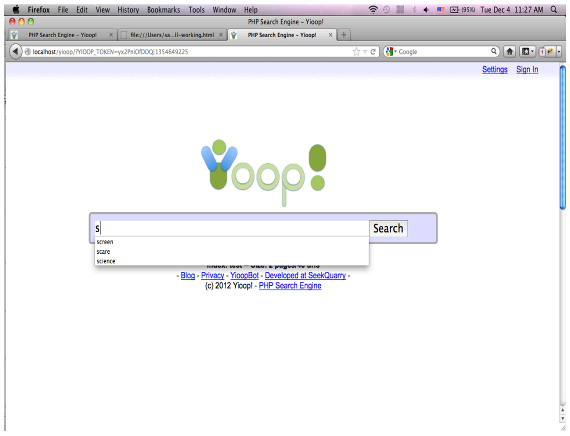
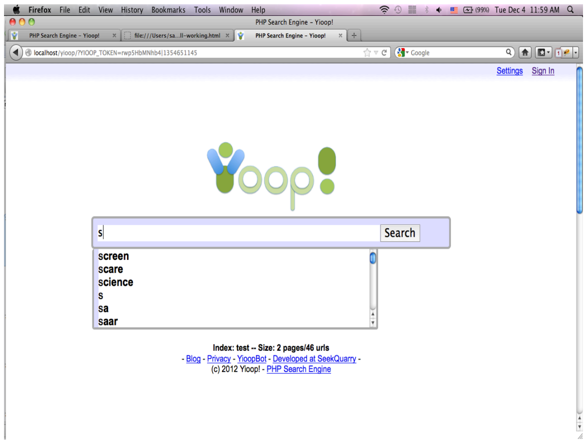

Settings
Sign In
Yioop!
s
Search
screen
scare
science
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Settings
Sign In
Yioop!
s
Search
screen
scare
science
s
sa
saar
Index: test -- Size: 2 pages/46 urls
- Blog - Privacy - YioopBot - Developed at SeekQuarry -
(c) 2012 Yioop! - PHP Search Engine

Figure 10.1 Comparison of usage of previous queries

In this scenario, both work in similar fashion by ordering the suggestions in the frequency of their usage. An addition to Yioop is that it also appends the results from the dictionary, which is quite helpful.

Foreign language support:

Foreign language queries are supported both by the browser and Yioop but the browser only suggests the previously typed queries. On the other hand autosuggest in Yioop uses both local storage and dictionary and provides a suggestion list.

Spell correction:

There is no spell correction in the browser. As discussed in detail, in the above sections, the query is corrected and suggested on the search results page. This saves the user time if the corrected word is actually what he intended, as he is just a click away to fire a search, rather than having to retype the query.

11. CONCLUSION & FUTURE WORK

In this project, the features, autosuggest and spell correction, have been implemented for Yioop, keeping in mind its constrained environment. Implemented in Javascript, autosuggest includes features like multi-word suggest, foreign language support and usage of locally stored previous queries, to enhance the performance. Spell correction was implemented for English language, assuming that the frequency of one-letter errors is more than multiple-letter errors. This was also a trade off made to avoid huge computations in Javascript, which would slower the performance of Yioop. These features proved to reduce the typing work and avoid spelling errors in Yioop.

In the future, autosuggest and spell correction features in Yioop can be extended and enhanced in following directions:

1. Introducing spelling correction for foreign languages like, French and Russian, which includes handling complex multi-byte characters.
2. Introducing suggestions for transliterated queries pertaining to languages other than Telugu.
3. If a user chooses to reuse the index multiple times, autosuggest can be enhanced by using the data from search results, where a trie can be constructed on the fly from the key words of search results.

12. REFERENCES

[1] Yioop website: www.yioop.com Retrieved Nov 30, 2012

[2] Autosuggest: http://en.wikipedia.org/wiki/Autocomplete Retrieved Nov 30, 2012

[3] Edit distance: http://en.wikipedia.org/wiki/Levenshtein_distance Retrieved Nov 30, 2012

[4] Telugu: http://en.wikipedia.org/wiki/Telugu_language Retrieved Nov 30, 2012

[5] Yioop documentation: http://www.seekquarry.com/?c=main&p=documentation Retrieved Dec 4, 2012

[6] Popular English words: http://books.google.com/ngrams/datasets Retrieved May 15, 2012

[7] Trie: http://en.wikipedia.org/wiki/Trie Retrieved May 15, 2012

[8] Stop words: http://en.wikipedia.org/wiki/Stop_words Retrieved Dec 3, 2012

[9] Firefox web console: https://developer.mozilla.org/en-US/docs/Tools/Web_Console Retrieved Nov 30, 2012

[10] Local storage: http://www.w3schools.com/html/html5_webstorage.asp Retrieved Nov 30, 2012

[11] Spell correction: http://norvig.com/spell-correct.html Retrieved Nov 30, 2012

[12] Telugu writing system: http://en.wikipedia.org/wiki/Telugu_language#Writing_system Retrieved Nov 30, 2012

[13] Transliteration based Text Input Methods For Telugu, V.B. Sowmya and Vasudeva Varma , *22and International Conference on Computer Processing for Oriental Languages" 2009.*

[14] Telugu dictionary: http://www.lib.uchicago.edu/e/su/southasia/to_vijay/gwynn.txt Retrieved Nov 30, 2012